```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdint.h>
#include "stm32f0xx.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim16;

/* USER CODE BEGIN PV */
// TODO: Define input variables
// general flags:
uint8_t timing = 1; // 1 for 1 second and 0 for 0.5 seconds
uint8_t mode; // mode is either 1, 2 or 3
// for mode 1 and 2:
uint16_t pattern;
uint8_t up_or_down = 0;
uint8_t curr_led = 0;
// for mode 36
uint32_t delay_hold;
```

```
uint32_t delay_switch_off;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM16_Init(void);
/* USER CODE BEGIN PFP */
void TIM16_IRQHandler(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM16_Init();
  /* USER CODE BEGIN 2 */

  // TODO: Start timer TIM16
  HAL_TIM_Base_Start_IT(&htim16); // start timer and enable interrupts



  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    // TODO: Check pushbuttons to change timer delay
    if(!LL_GPIO_IsInputPinSet(Button1_GPIO_Port, Button1_Pin)){
        mode = 1;
    }else if(!LL_GPIO_IsInputPinSet(Button2_GPIO_Port, Button2_Pin)){
        mode = 2;
    }else if(!LL_GPIO_IsInputPinSet(Button3_GPIO_Port, Button3_Pin)){
        mode = 3;
    }else if(!LL_GPIO_IsInputPinSet(Button0_GPIO_Port, Button0_Pin)){
        // if timing is 1 then change the ARR such that it counts to 0.5 seconds and if timing is 0, change it such that it counts to 1 second
        if(timing == 1){
            TIM16->ARR = 499; // set ARR to count from 0 to 499 so it counts for 500 ticks of 1 ms each to make 0.5 seconds per overflow
          timing = 0; // acknowledge change with flag
        }else if(timing == 0){
            TIM16->ARR = 999; // set ARR so it counts to 1 second
            timing = 1; // acknowledge change with flag
        }
    }

   }
   /* USER CODE END 3 */
  }
```

```
/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
 LL_FLASH_SetLatency(LL_FLASH_LATENCY_0);
 while(LL_FLASH_GetLatency() != LL_FLASH_LATENCY_0)
 {
 }
 LL_RCC_HSI_Enable();

  /* Wait till HSI is ready */
 while(LL_RCC_HSI_IsReady() != 1)
 {

 }
 LL_RCC_HSI_SetCalibTrimming(16);
 LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
 LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_1);
 LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_HSI);

  /* Wait till System clock is ready */
 while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_HSI)
 {

 }
```

```
  LL_SetSystemCoreClock(8000000);

   /* Update the time base */
  if (HAL_InitTick (TICK_INT_PRIORITY) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM16 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM16_Init(void)
{

  /* USER CODE BEGIN TIM16_Init 0 */

  /* USER CODE END TIM16_Init 0 */

  /* USER CODE BEGIN TIM16_Init 1 */

  /* USER CODE END TIM16_Init 1 */
  htim16.Instance = TIM16;
  htim16.Init.Prescaler = 8000-1;
  htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim16.Init.Period = 1000-1;
```

```
 htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
 htim16.Init.RepetitionCounter = 0;
 htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
 if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
 {
  Error_Handler();
 }
 /* USER CODE BEGIN TIM16_Init 2 */
 NVIC_EnableIRQ(TIM16_IRQn); // add TIM16 to NVIC
 /* USER CODE END TIM16_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
 LL_GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

 /* GPIO Ports Clock Enable */
 LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOF);
 LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
 LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);
```

```
/**/

LL_GPIO_ResetOutputPin(LED0_GPIO_Port, LED0_Pin);

/**/

LL_GPIO_ResetOutputPin(LED1_GPIO_Port, LED1_Pin);

/**/

LL_GPIO_ResetOutputPin(LED2_GPIO_Port, LED2_Pin);

/**/

LL_GPIO_ResetOutputPin(LED3_GPIO_Port, LED3_Pin);

/**/

LL_GPIO_ResetOutputPin(LED4_GPIO_Port, LED4_Pin);

/**/

LL_GPIO_ResetOutputPin(LED5_GPIO_Port, LED5_Pin);

/**/

LL_GPIO_ResetOutputPin(LED6_GPIO_Port, LED6_Pin);

/**/

LL_GPIO_ResetOutputPin(LED7_GPIO_Port, LED7_Pin);

/**/

GPIO_InitStruct.Pin = Button0_Pin;

GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button0_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = Button1_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button1_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = Button2_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button2_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = Button3_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button3_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED0_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED0_GPIO_Port, &GPIO_InitStruct);
```

```
/**/
GPIO_InitStruct.Pin = LED1_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED1_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED2_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED2_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED3_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED3_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED4_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;

GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;

GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;

LL_GPIO_Init(LED4_GPIO_Port, &GPIO_InitStruct);


/**/

GPIO_InitStruct.Pin = LED5_Pin;

GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;

GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;

GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;

GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;

LL_GPIO_Init(LED5_GPIO_Port, &GPIO_InitStruct);


/**/

GPIO_InitStruct.Pin = LED6_Pin;

GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;

GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;

GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;

GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;

LL_GPIO_Init(LED6_GPIO_Port, &GPIO_InitStruct);


/**/

GPIO_InitStruct.Pin = LED7_Pin;

GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;

GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;

GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;

GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;

LL_GPIO_Init(LED7_GPIO_Port, &GPIO_InitStruct);
```

```
/* USER CODE BEGIN MX_GPIO_Init_2 */

/* USER CODE END MX_GPIO_Init_2 */

}


/* USER CODE BEGIN 4 */

void TIM16_IRQHandler(void)

{

	// Acknowledge interrupt

	HAL_TIM_IRQHandler(&htim16);


	// TODO: Change LED pattern

  if((mode == 1) || (mode == 2)){

    if((up_or_down == 1)&&(curr_led != 7)){ // moving up

      curr_led++;

      pattern = 1 << curr_led; // value that in binary will be the next digit (led) only

    }else if((up_or_down == 0)&&(curr_led != 0)){ // moving down

      curr_led--;

      pattern = 1 << curr_led;

    }else if(curr_led == 7){ // at the top

      up_or_down = 0; // turn around and move down

      curr_led = 6; // move to second from top led

      pattern = 1 << curr_led;

    }else if(curr_led == 0){ // at the bottom

      up_or_down = 1; // turn around and move up

      curr_led = 1; // move to second from bottom led

      pattern = 1 << curr_led;

    }
```

```
      if(mode == 1){
          GPIOB->ODR = pattern; // display pattern which is 1 led lighting up by itself
      }else{
          GPIOB->ODR = ~(pattern); // same as mode 1 but reversed (only 1 led is off at a time)
      }

    }else if(mode == 3){
          // get random value to display and random delay values
          delay_hold = rand()%(1500-100+1) + 100; // delay to hold for initially (between 100 and 500 milliseconds)
          pattern = rand()%256; // value to display (when converted to binary this will correspond to which leds turn on)
          // set output
          GPIOB->ODR = pattern;
          // delay
      delay(delay_hold);

          // pseudo-random order to check leds status and turn them off 'randomly'
          // delay of a random amount from 0 to 100 milliseconds between switch-off events of leds
          if(LL_GPIO_IsInputPinSet(LED3_GPIO_Port, LED3_Pin)){
                  LL_GPIO_ResetOutputPin(LED3_GPIO_Port, LED3_Pin);
                  delay_switch_off = rand()%101; // delay value for between switch-offs (between 0 and 100 milliseconds)
                  delay(delay_switch_off);
          }

          if(LL_GPIO_IsInputPinSet(LED5_GPIO_Port, LED5_Pin)){
                  LL_GPIO_ResetOutputPin(LED5_GPIO_Port, LED5_Pin);
                  delay_switch_off = rand()%101;
                  delay(delay_switch_off);
          }

          if(LL_GPIO_IsInputPinSet(LED7_GPIO_Port, LED7_Pin)){
            LL_GPIO_ResetOutputPin(LED7_GPIO_Port, LED7_Pin);
            delay_switch_off = rand()%101;
            delay(delay_switch_off);
      }

          if(LL_GPIO_IsInputPinSet(LED0_GPIO_Port, LED0_Pin)){
            LL_GPIO_ResetOutputPin(LED0_GPIO_Port, LED0_Pin);
            delay_switch_off = rand()%101;
            delay(delay_switch_off);
      }

          if(LL_GPIO_IsInputPinSet(LED4_GPIO_Port, LED4_Pin)){
            LL_GPIO_ResetOutputPin(LED4_GPIO_Port, LED4_Pin);
            delay_switch_off = rand()%101;
            delay(delay_switch_off);
          }

          if(LL_GPIO_IsInputPinSet(LED6_GPIO_Port, LED6_Pin)){
            LL_GPIO_ResetOutputPin(LED6_GPIO_Port, LED6_Pin);
            delay_switch_off = rand()%101;
            delay(delay_switch_off);
          }
```

```
        if(LL_GPIO_IsInputPinSet(LED2_GPIO_Port, LED2_Pin)){
          LL_GPIO_ResetOutputPin(LED2_GPIO_Port, LED2_Pin);
          delay_switch_off = rand()%101;
          delay(delay_switch_off);
        }

        if(LL_GPIO_IsInputPinSet(LED1_GPIO_Port, LED1_Pin)){
          LL_GPIO_ResetOutputPin(LED1_GPIO_Port, LED1_Pin);
          delay_switch_off = rand()%101;
          delay(delay_switch_off);
        }

  }



}


/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
/**
 * Basic delay function for mode 3
 */
void delay(uint32_t delay_time){
	uint32_t i = (delay_time*8000)/8;
	uint32_t i_in = i/1000;
	for(volatile uint32_t j=0; j<1000; j++){
	  for(volatile uint32_t m=0; m<(i_in); m++){

	  }
	}
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
```

```
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```